Formal Verification via MCMAS & PRISM

Hongyang Qu

University of Sheffield

1 December 2015

Outline

- Motivation for Formal Verification
- Overview of MCMAS
- Overview of PRISM

Verification

Developer

Verification Agent

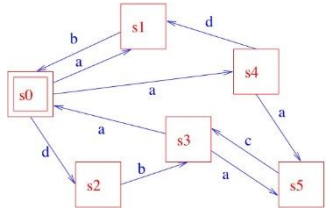
Formal verification

- It is a systematic way to check all behaviour of a system with respect to certain specification

System

Abstraction

Mathematical model



Specification

Abstraction

Logic formula

$$\Box((call \vee \Diamond open) \rightarrow ((\neg atfloor \vee \neg open)\ \mathcal{U}\ (open \vee ((atfloor \wedge \neg open)\ \mathcal{U}\ (open \vee ((\neg atfloor \wedge \neg open)\ \mathcal{U}\ (open \vee ((atfloor \wedge \neg open)\ \mathcal{U}\ (open \vee (\neg atfloor\ \mathcal{U}\ open)))))))))$$

Verification algorithm

Result

Why formal verification is important?

Pentium FDIV bug

$475 million loss

Replacing testing with formal verification

The Explosion of the Ariane 5

$500 million loss

Logic verification of critical subsystems

Can driverless cars run politely?

An example in robotics

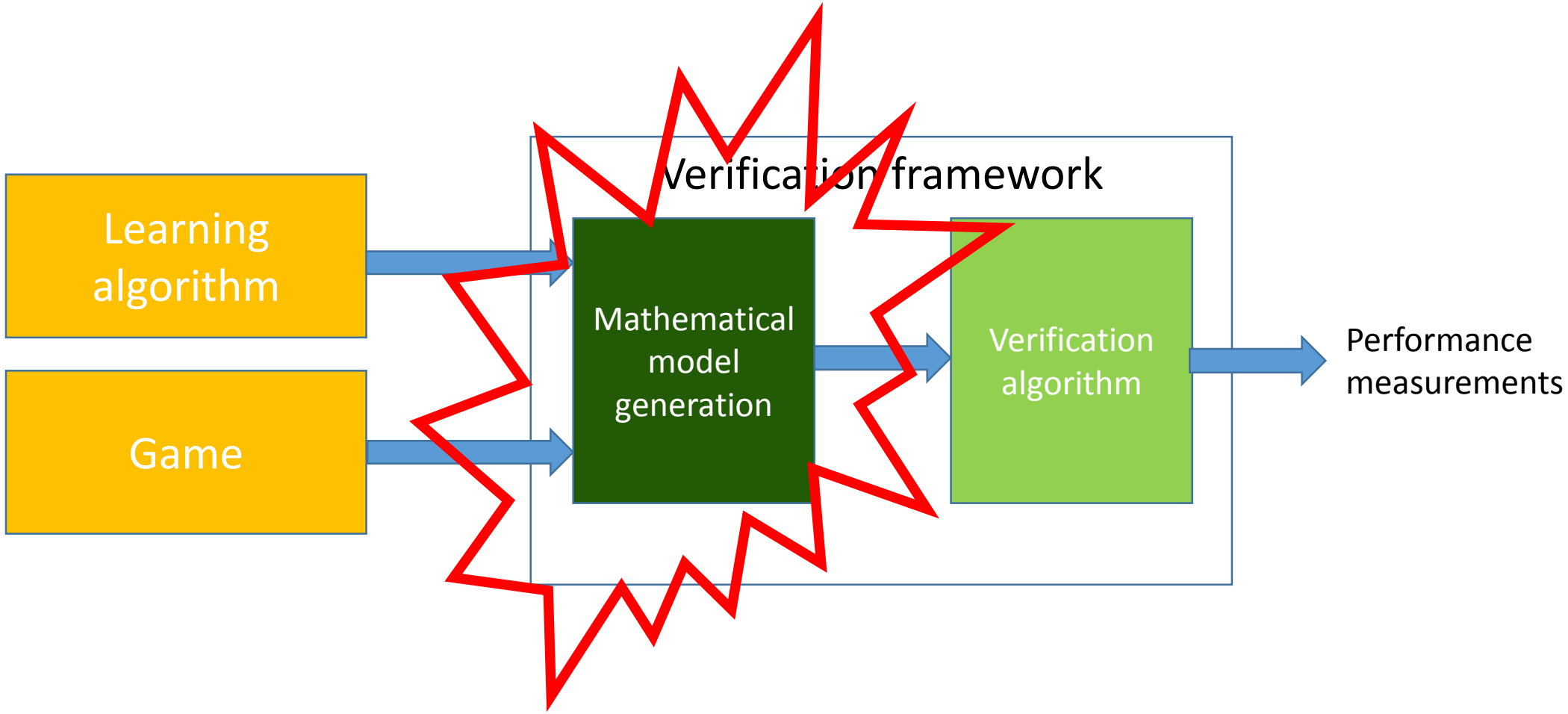
- Two UAVs fly towards each other at the same altitude
  - Each UAV has two actions:

    High altitude and low altitude
  - UAV 1: action A (high) or B (low)
  - UAV2: action C (low) and D (high)

	Action C	Action D
Action A	✓	×
Action B	×	✓

This scenario can be cast as a game

Verification framework for analysing performance of learning algorithms



MCMAS: A Model Checker for Multi-Agent Systems

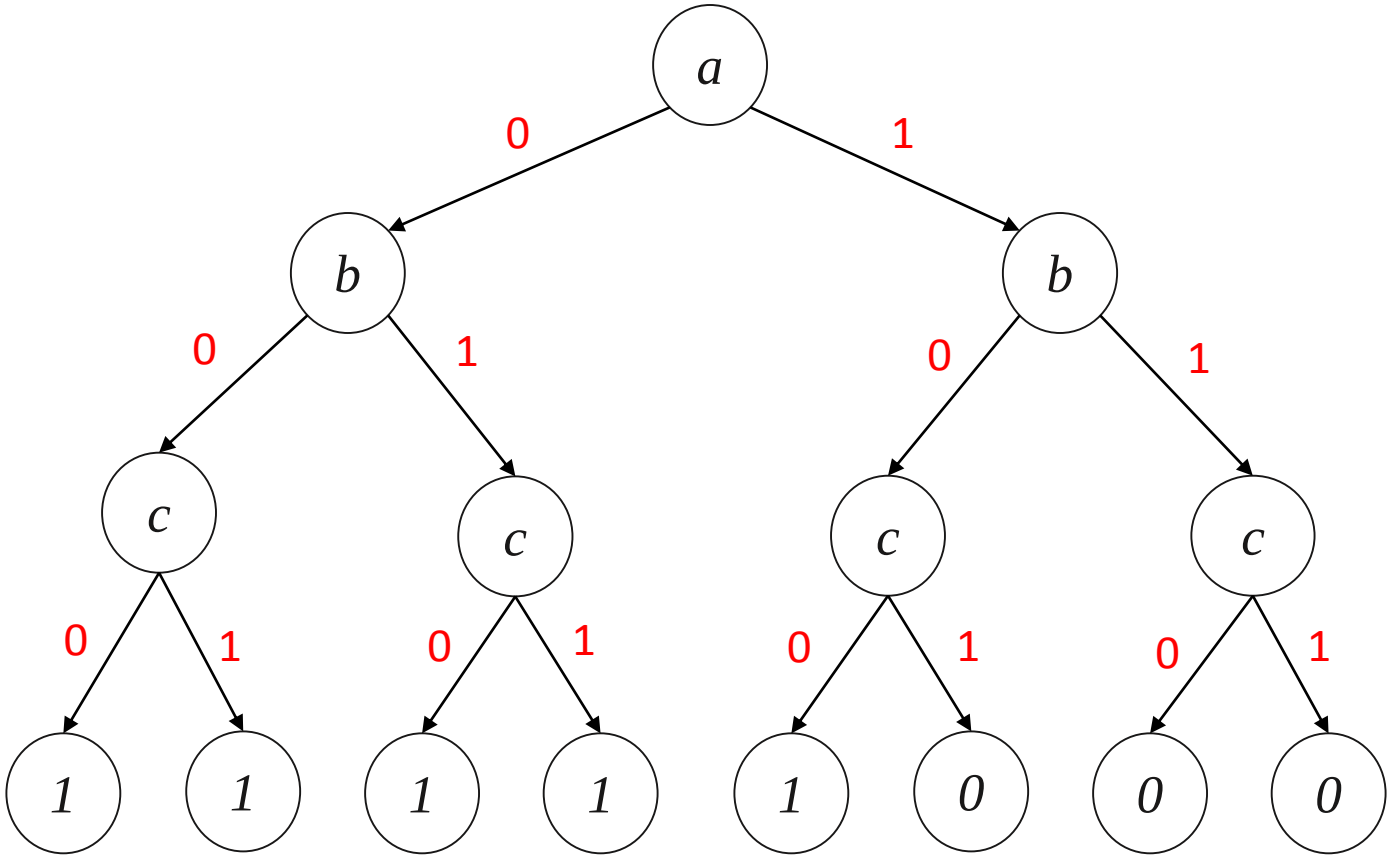
- Multi-agent systems are an active research area in Artificial Intelligence (AI).
  - They can be used to solve problems that are difficult or impossible for an individual agent or a monolithic system to solve.

- MCMAS can check complex properties, generate executions leading to bugs and find strategies for game models.

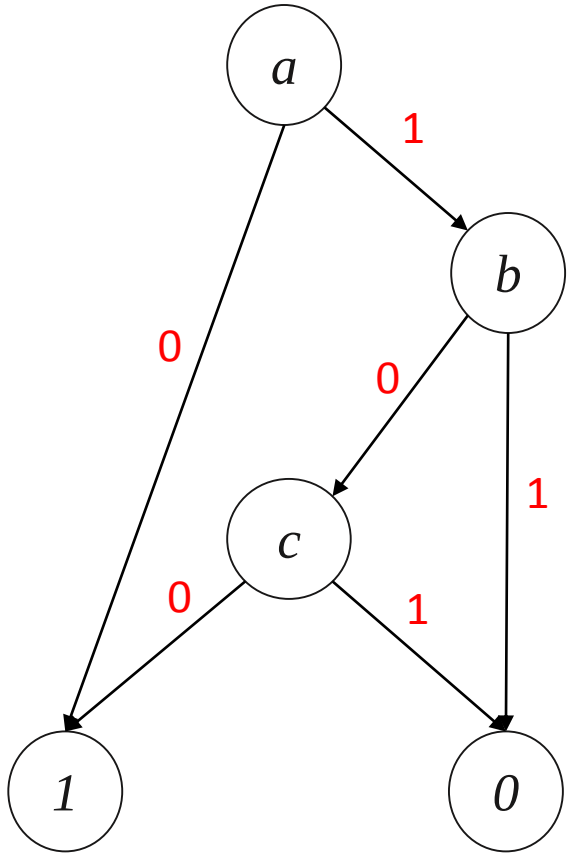
MCMAS (http://vas.doc.ic.ac.uk/software/mcmas/)

- Symbolic model checker via OBDDs
- Input language ISPL (Interpreted Systems Programming Language)
- Support CTL + Epistemic logic + ATL
- Support (unconditional) fairness
- Efficient implementation of model checking algorithms
- Counterexample/witness generation
- Eclipse-based GUI
- Many applications
- It is actively maintained and developed.

Ordered Binary Decision Diagram (OBDD)



Truth table of $\neg a \vee (a \wedge \neg b \wedge \neg c)$



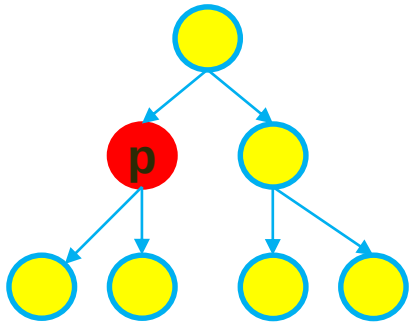
OBDD of $\neg a \vee (a \wedge \neg b \wedge \neg c)$

Interpreted systems

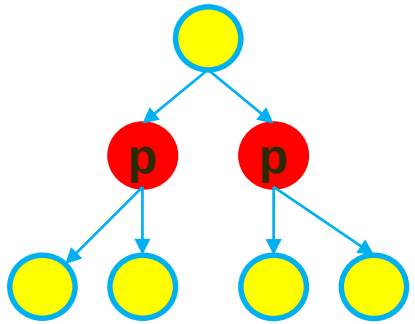
- An interpreted system IS is composed of N agents $A = \{1, \dots, n\}$
- Each agent $i \in A$ has
  - a finite set of *local states* $L_i = \left\{l_i^1, \dots, l_i^{nl_i}\right\}$ and
  - a finite set of *actions* $Act_i = \left\{a_i^1, \dots, a_i^{na_i}\right\}$
  - a *local protocol* $P_i : L_i \to 2^{Act_i}$
  - an *evolution function* $Ev_i : L_i \times Act_1 \times \cdots \times Act_n \to L_i$
- A global state is $s = (l_1, \dots, l_n)$, and the set of states is S
- A global joint action is $a = (a_1, \dots, a_n)$

Computation Tree Logic (CTL)

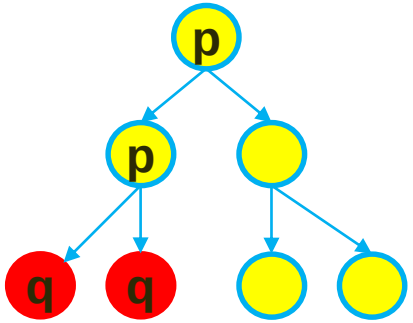
- $\varphi ::= p|\neg\varphi|\varphi \wedge \varphi$
  $|EX\varphi|EG\varphi|EF\varphi|E(\varphi U\varphi)$
  $|AX\varphi|AG\varphi|AF\varphi|A(\varphi U\varphi)$
  - Path quantifier:
    E (exists) and A (all)
  - Temporal operator:
    X (next), G (globally),
    U (until) and F (future)



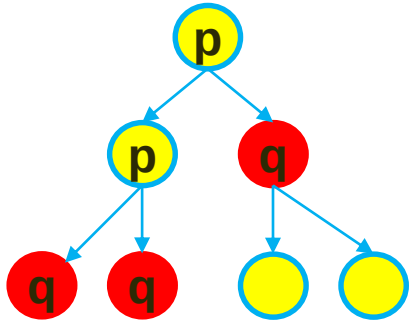
EXp



AXp



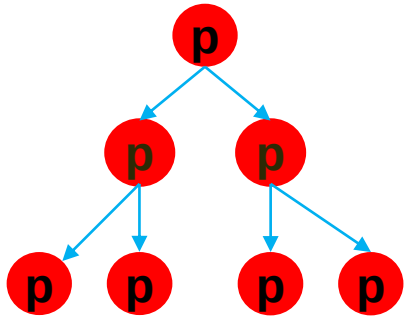
$E(pUq)$



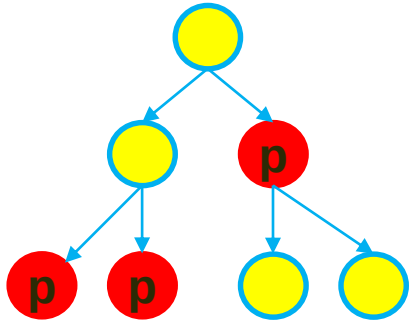
$A(pUq)$

Computation Tree Logic (CTL)

- $\varphi ::= p|\neg\varphi|\varphi \wedge \varphi$
  $|EX\varphi|EG\varphi|EF\varphi|E(\varphi U\varphi)$
  $|AX\varphi|AG\varphi|AF\varphi|A(\varphi U\varphi)$
  - Path quantifier:
    E (exists) and A (all)
  - Temporal operator:
    X (next), G (globally),
    U (until) and F (future)

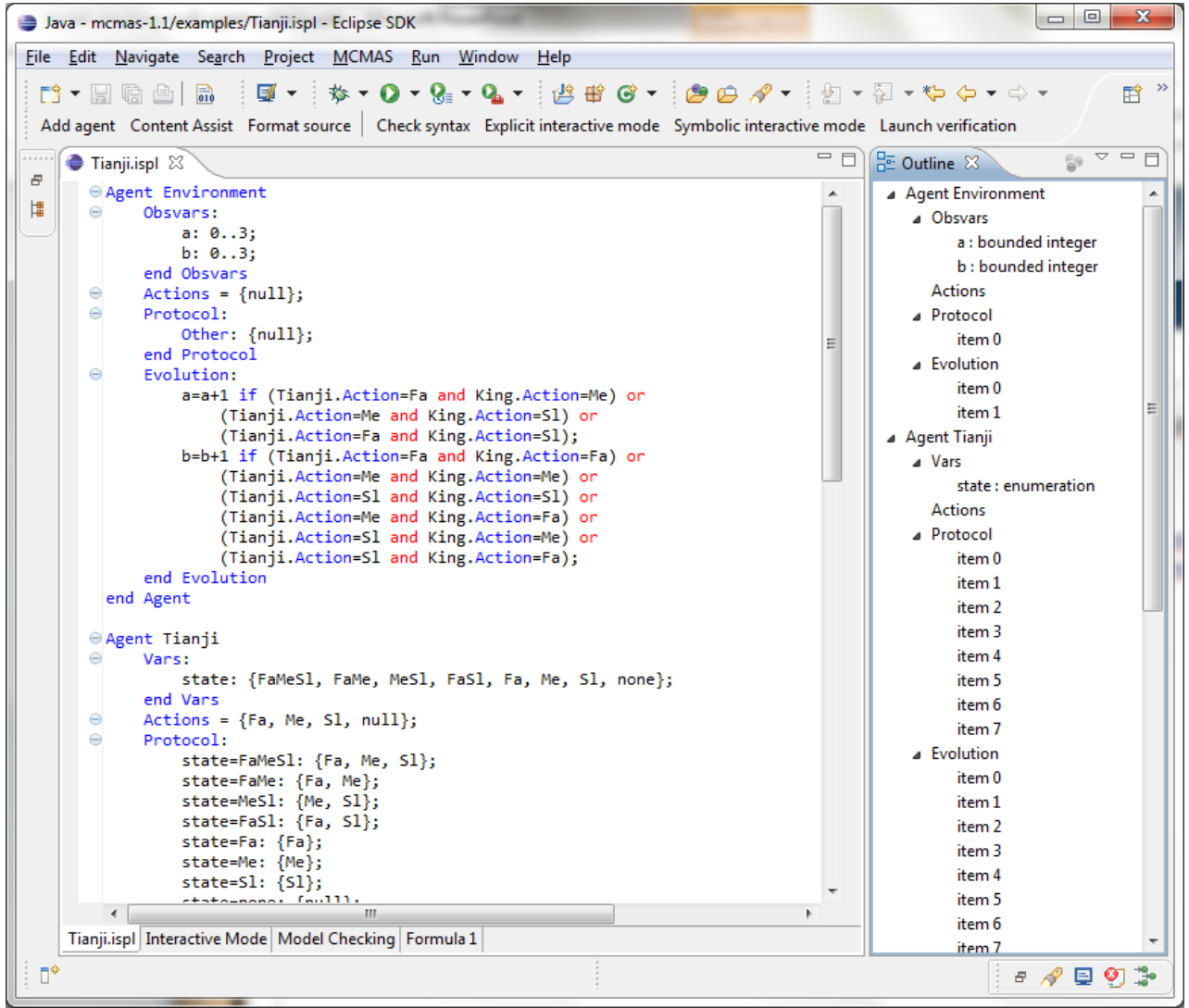


AGp

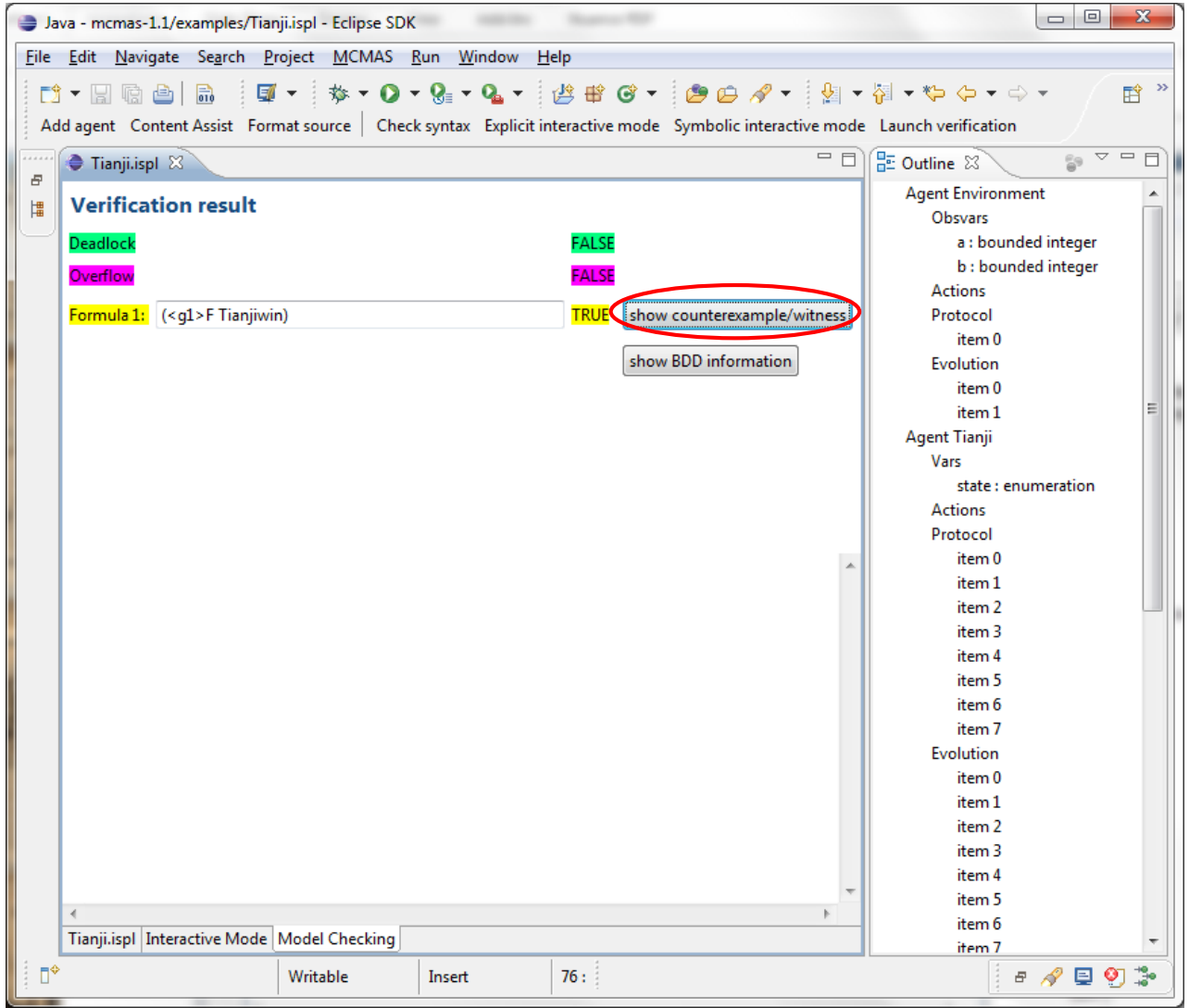


AFp

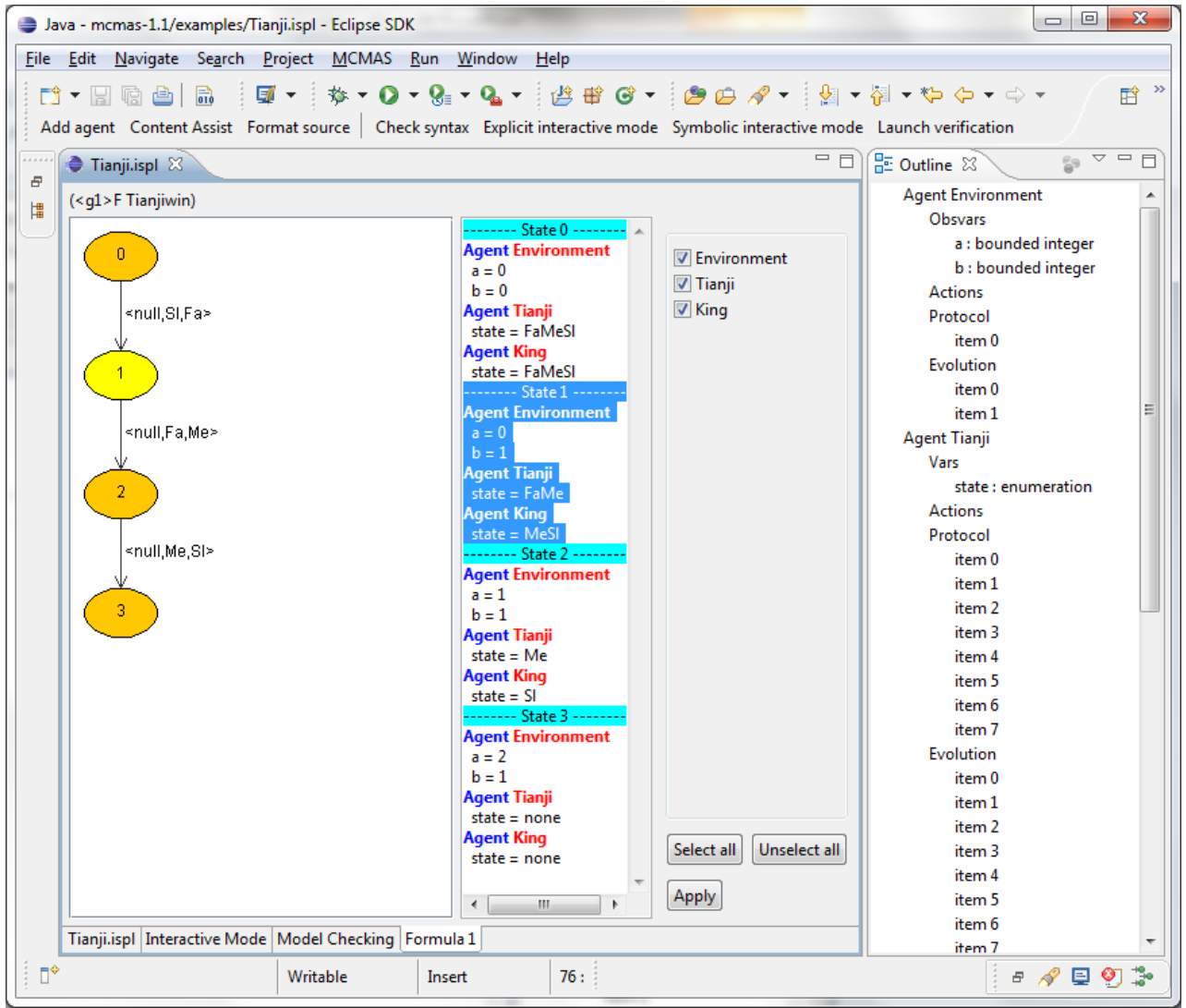
MCMAS screenshots (1)



MCMAS screenshots (2)



MCMAS screenshots (3)



Case study: Inconsistent reasoning

- A robot has one sensing event and two decision predicates
  - a: sensing event
  - b, c: predicates
- Reasoning rules:
  - $a \rightarrow \neg b$
  - $a \rightarrow c$
  - $\neg b \rightarrow \neg c$
- Initially, a is true, b and c are unknown

MCMAS model (1)

```
Semantics = SingleAssignment;
Agent M
   Vars:
      a: boolean;
      b: {unknown, TRUE, FALSE};
      c: {unknown, TRUE, FALSE};
   end Vars
   Actions = {none};
   Protocol:
      Other: {none};
   end Protocol
   Evolution:
      b=FALSE if a=true;
      c=TRUE if a=true;
     b=FALSE if c=FALSE;
   end Evolution
end Agent
```

```
Evaluation
   a_true if M.a=true;
   a_false if M.a=false;

   b_true if M.b=TRUE;
   b_false if M.b=FALSE;
   b_unknown if M.b=unknown;

   c_true if M.c=TRUE;
   c_false if M.c=FALSE;
   c_unknown if M.c=unknown;
end Evaluation

InitStates
   M.a=true and M.b=unknown and M.c=unknown;
end InitStates
```

MCMAS model (2)

Formulae

AF (((AG a_true) or (AG a_false)) and
((AG b_true) or (AG b_false) or (AG b_unknown)) and
((AG c_true) or (AG c_false) or (AG c_unknown)));

AG ((!((EX a_true) and (EX a_false))) and
(!((EX b_true) and (EX b_false))) and
(!((EX c_true) and (EX c_false))));

end Formulae

Formula 1: Eventually all variables won't change their value (become stable)

Formula 2: It is always that no variable can be assigned to different values.

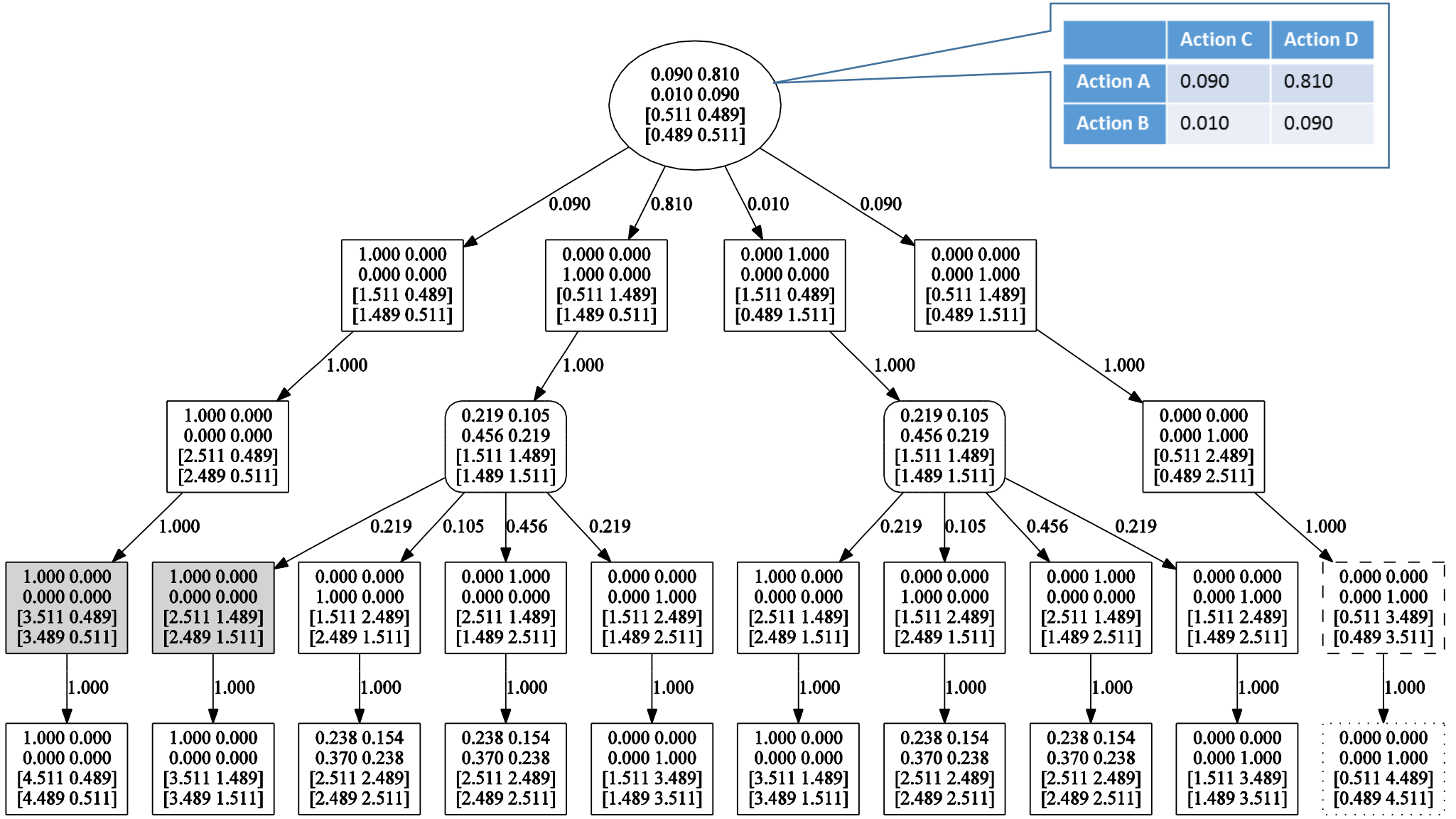
PRISM (http://www.prismmodelchecker.org/)

- The most popular probabilistic model checker for verifying/analysing systems that have probabilistic behaviour
  - Support rich probabilistic models and specification languages
  - Various verification engines (MTBDD, sparse, hybrid, explicit)
  - State-of-the-art performance
  - Intuitive GUI
  - Actively maintained and developed
  - Has been applied to analyse swarm robots, robot coordination, autonomous systems, and many others.

Discrete-Time Markov Chains (DTMCs)

- A DTMC is a state-transition system with transitions labelled probabilities
  - A state is a possible configuration of the system
  - Transitions between states represent evolution of the system
  - From a state, the system can move to other states with certain probabilities
- Can be represented as a tuple $M = (S, \mathrm{Steps}, \bar{s})$ where
  - S is a finite set of states
  - $\bar{s} \in S$ is the initial state
  - $\mathrm{Steps}: S \to Dist(S)$ is a probabilistic transition function
- A DTMC is memoryless, which means the probability distribution in a state does not depend on the history of evolution

DTMC model for coordination between UAVs



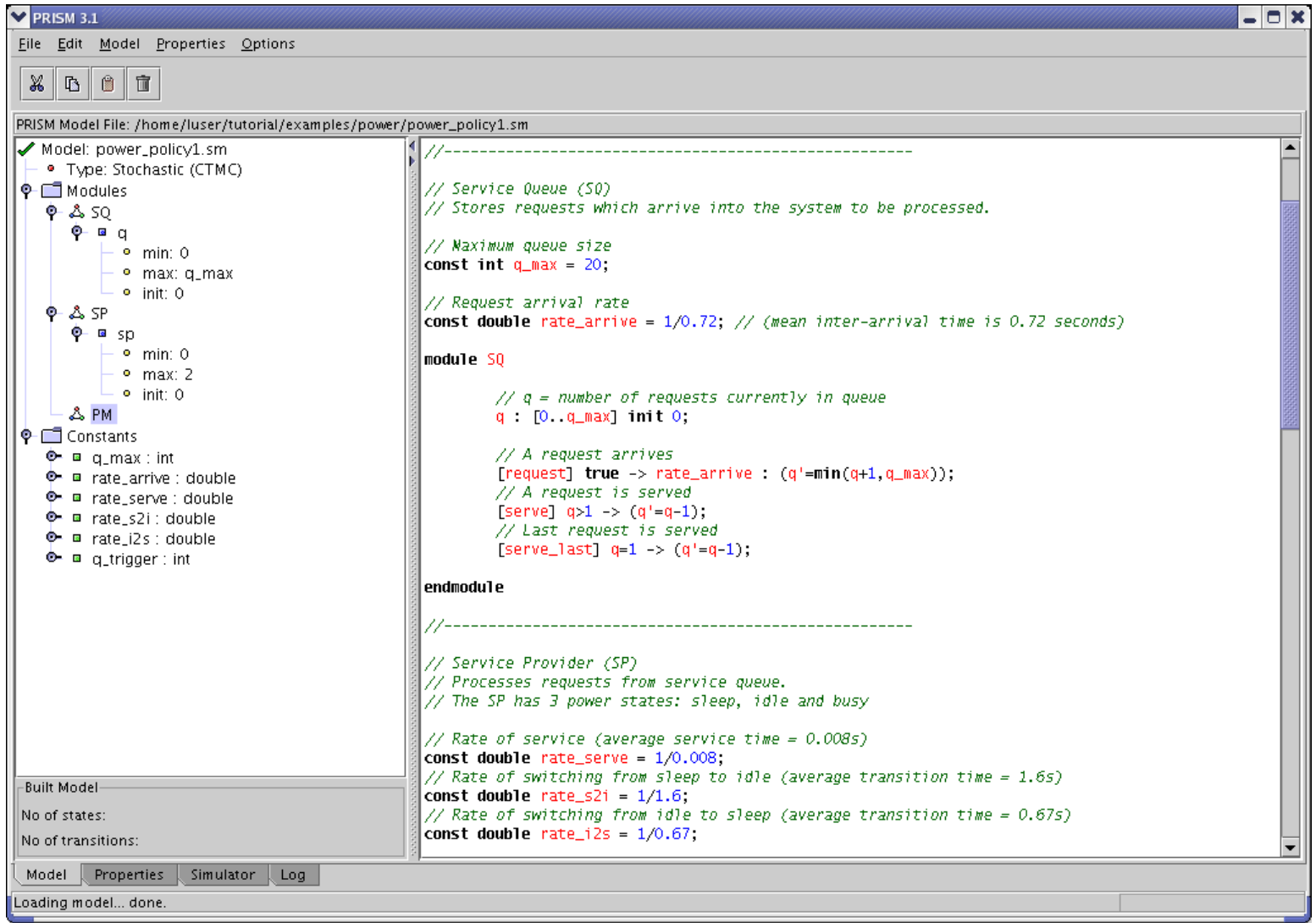
Other porpular probabilistic models

- Markov Decision Processes (MDP)
  - $M = (S, \Sigma, \mathrm{Steps}, \bar{s})$ where
    - Σ is a finite set of actions
    - $\mathrm{Steps}: (S \times \Sigma) \to Dist(S)$ is a probabilistic transition function
- Continuous-Time Markov Chains (CTMC)
  - $M = (S, R, \bar{s})$ where
    - $R: S \times S \to \mathbb{R}_{>0}$ is a transition rate matrix

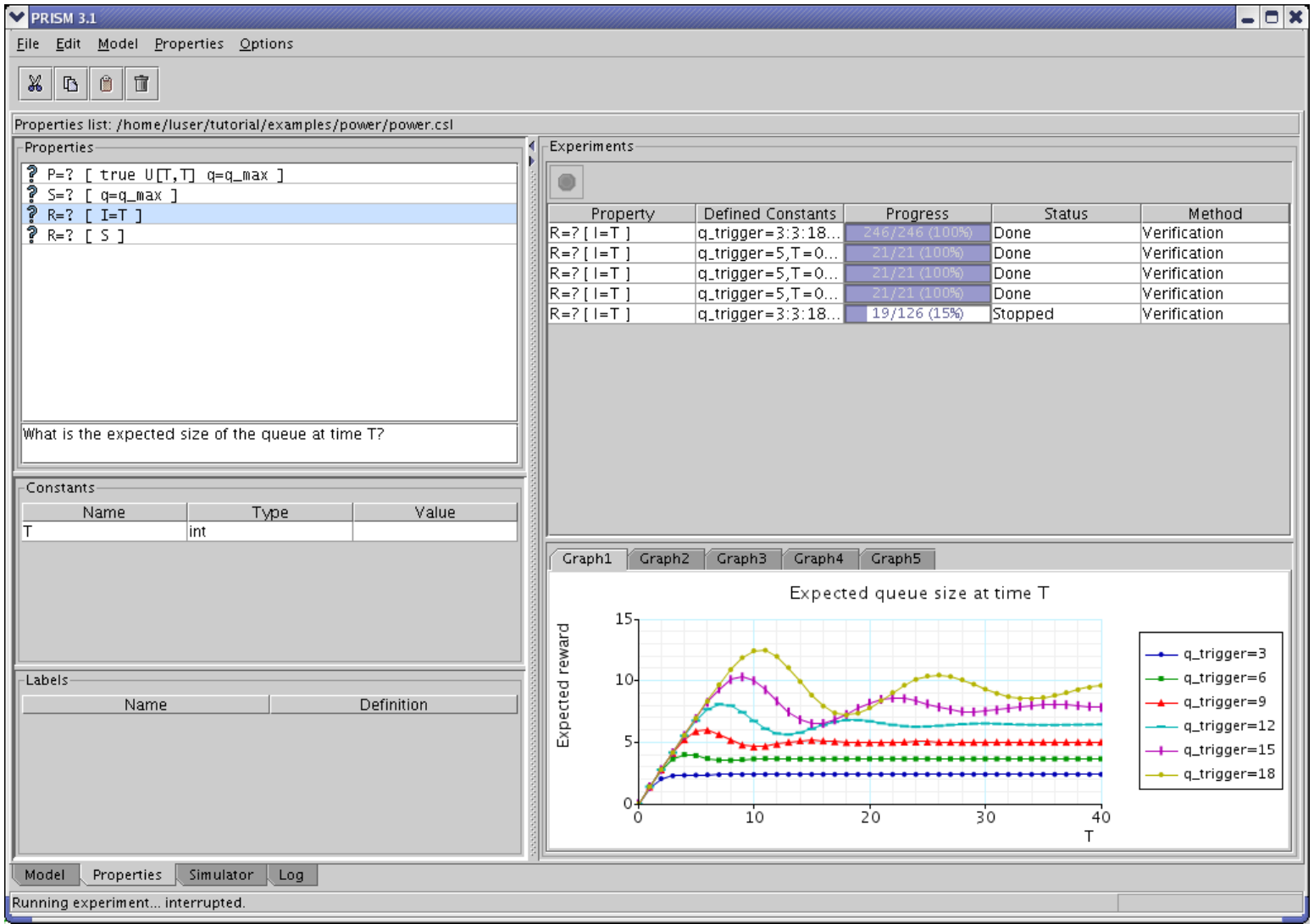
Probabilistic Specifications

- Reachability properties
  - The probability of reaching a set of states from the initial state
  - Example: A message is delivered successfully with probability 90%.
- Steady state properties
  - The probability of staying in a state (Nash equilibrium) in the long run
  - Example: What is the probability of the queue being 50% full in the long run?
- Reward properties
  - Properties about instantaneous/cumulative rewards attached to states and/or transitions
  - Example: What is the average elapse time of delivering a message?
- Verification of probabilistic properties involves heavy matrix operations (usually multiplications)

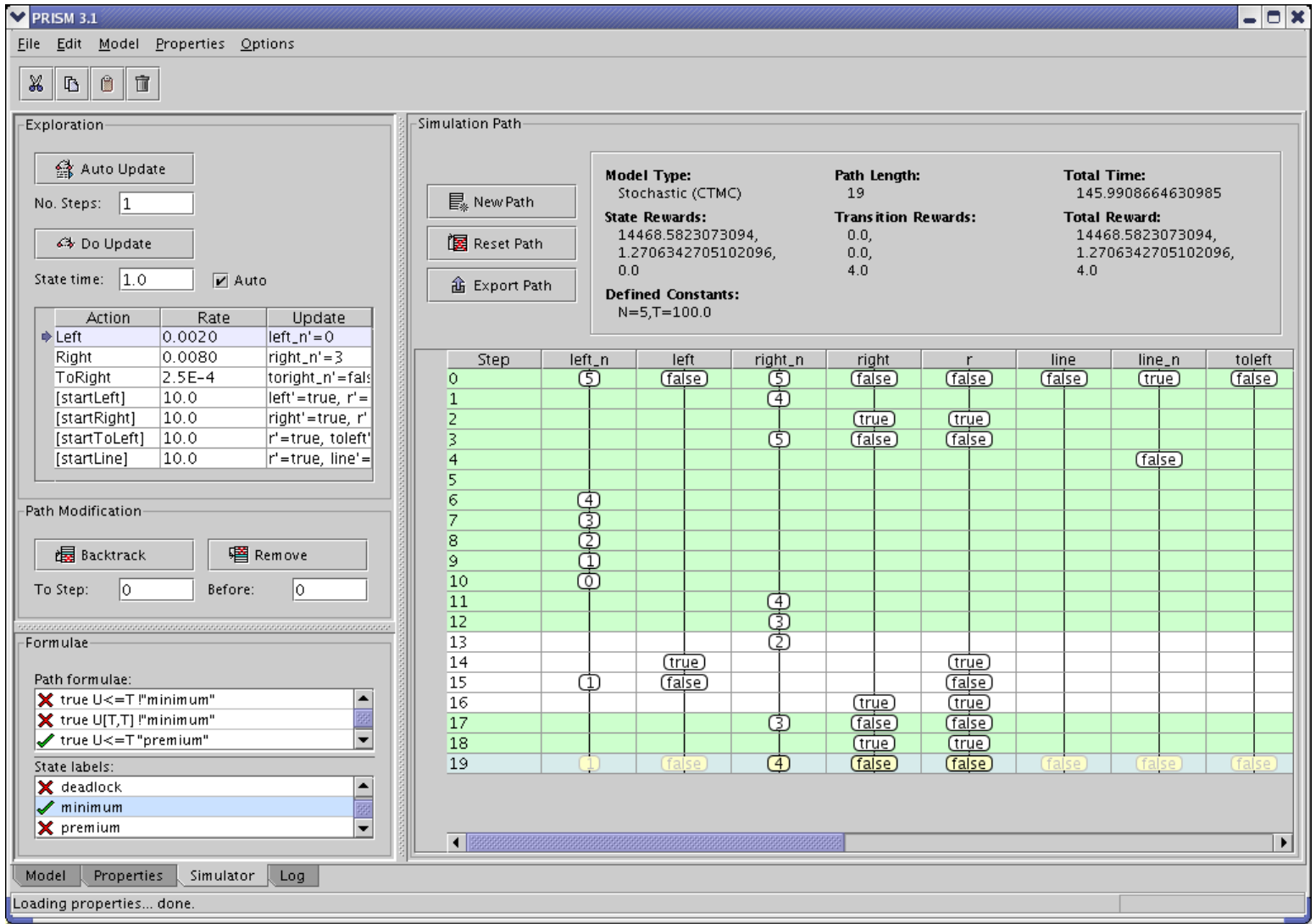
PRISM screenshots (1)



PRISM screenshots (2)

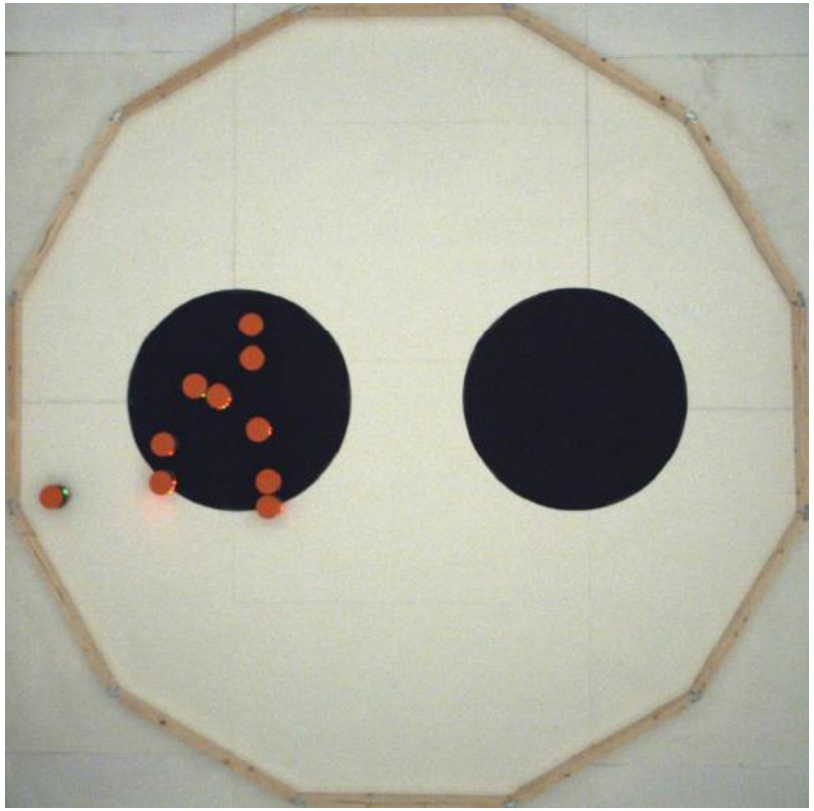


PRISM screenshots (3)



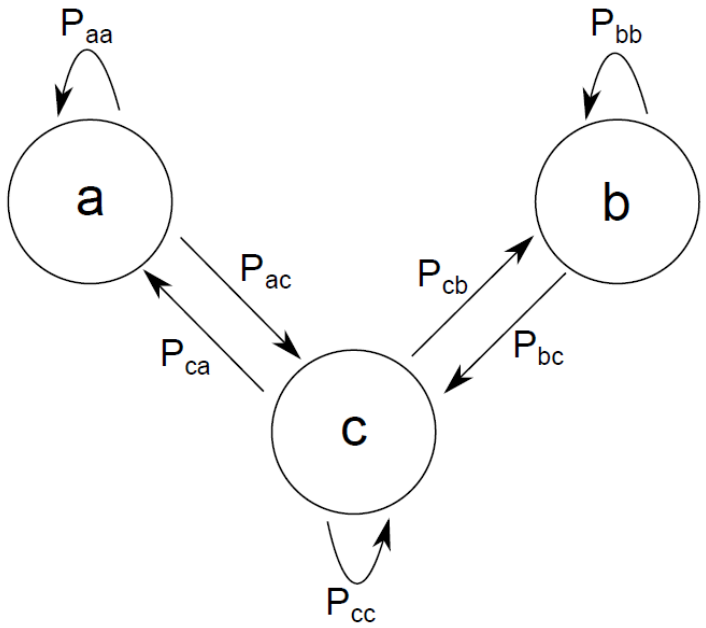
Case study: swarm aggregation

- The robots have to cluster in one of the two aggregation areas
- The robots go around at random and stop if they encounter a black spot (aggregation area)
- According to a certain probability, they leave the aggregation area and restart walking randomly



DTMC model

- $p_{ca} = p_{cb} = \frac{S_{agg}}{S_{all}}$
- $p_{aa} = 1 - p_{ac}, p_{bb} = 1 - p_{bc}, p_{cc} = 1 - p_{ca} - p_{cb}$
- $p_{ac} = p_{bc} = p_{max} \times (1 - \frac{N_S}{N})$



PRISM program (1)

```
dtmc
const int N = 3;
const double Pca = 0.08;
const double Pcb = Pca;
const double P_max = 0.2;
formula Pac = P_max * (1 - a/N);
formula Pbc = P_max * (1 - b/N);
```

PRISM program (2)

```
module robots
  a : [0..N] init 0;
  b : [0..N] init 0;
  c : [0..N] init N;
  [] true -> c/N*Pca: (a'=min(a+1,N))&(c'=max(c-1,0)) +
                     c/N*Pcb: (b'=min(b+1,N))&(c'=max(c-1,0)) +
                     a/N*Pac: (a'=max(a-1,0))&(c'=min(c+1,N)) +
                     b/N*Pbc: (b'=max(b-1,0))&(c'=min(c+1,N)) +
                     (1-c/N*Pca-c/N*Pcb-a/N*Pac-b/N*Pbc): true;

endmodule
```

Probabilistic properties

- Let " areaA " = $a = N$ and "areaB" = $b = N$;
  - P=? [F "areaA"|"areaB"]

    What is the probability of all robots entering area A or area B?

  - S=? ["areaA"]

    In the long run, what is the probability of all robots staying in area A?

References

- **Alessio Lomuscio, Hongyang Qu, Franco Raimondi.** MCMAS: An open-source model checker for the verification of multi-agent systems. International Journal on Software Tools for Technology Transfer (STTT), 2015
- **Marta Kwiatkowska, Gethin Norman and David Parker.** PRISM 4.0: Verification of Probabilistic Real-time Systems. In *Proc. 23rd International Conference on Computer Aided Verification (CAV'11)*, volume 6806 of LNCS, pages 585-591, 2011.